Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.4 No.11 August 1984

The KAOS stand at the recent PC-84 Computer show which was held at the World Trade Centre in Melbourne.

Above:
John Lemcke (seated) and Jeff Rae.

Photos by M. McDonald

The next meeting will be held at 2pm on Sunday 26th August 1984 at the Essendon Primary School on the corner of Raleigh and Nicholson Streets, Essendon. The school will be open at 1 pm.

Closing date for articles for the September newsletter is 14th September.

INDEX

THE MEETING WAS KAOS

by King Corky

The meeting was a short one and not a lot of new info surfaced. Paul Dodd is still pushing the Compsoft CP/M cards for the Rabble/OSI's and did say that more OSI users are needed to mount the extension if any meaningful software development is to take place. I believe that even though $45Ø.ØØ is not chicken feed, the system represents extremely good valve for money when you consider that your OSI or Rabble system will be capable of accessing almost any format 8 bit CP/M, MSDOS, C, etc, software on either mini or maxi floppies, concurrently. If I hadn't blown my bankroll on a super-cheap Apple clone, I would have had the system installed in the 'monster' by now. I know that Ralph Hess has one running in his Rabble and is in the process of converting a lot of his software over to the CP/M system.

David Anear and Jeff Rae are considering setting up a machine language course for the 65Ø2. As far as we know, there is only one school in the state that teaches anything on the 65Ø2, Footscray Tech. David and Jeff are going to need some feedback and some help if the project is to get off the ground. Speaking of projects, David also suggested that something the club is badly lacking at present is hardware and software projects. Something that can be developed by a team. Suggestions were: a solderless breadboard that will plug into the 16/18/2Ø pin I/O, which will be compatable with the OSI, Rabble and possible even other 65Ø2 systems like the Apple, BBC, Atari, etc. This board could be like a flexible port for a variety of I/O activities ... A team to convert Apple (or any other) educational software to the OSI and Rabbles. There is a mountain of well written, fun to use, educational software available for the Apple, most of it with exceptional graphics which enhance the learning-by-computer type of program.

Ray Gardiner is looking for anyone interested in running hardware based floating-point maths through the 16 pin I/O. Ray also gave a brief rundown on the PC-84 show. Along with Ray, I think that anybody who attended would agree that the User Group stands were a brilliant idea. But more on that later. John Whitehead would like all cassette users to know that they can run Forth without having to upgrade to disk. He has a version configured for 24x24, 48x32, 64x32 and Cegmon systems and once he gets to know Forth a little better, he should be able to configure it to suit any format. Greg Kilfoyle, a past member, used his considerable talents to convert the once disk based language to run on cassette and we thank him most sincerely for his efforts.

And now my views of the PC-84 Computer Show. This outstanding expo, held at the World Trade Centre, simply put, was great. Of course the big guns were there, with their big flashy stands, but two things made this show stand out from any other of its type, including audio, car etc.

1. The marked absence of thousands of tiny, poking, game playing fingers, (there were still quite a few, but they didn't seem to hamper the prying eyes, ears and hands of those truly interested).

2. The amount of actual, fair dinkum help that the various exhibitors and their reps were prepared to give. Sure they were trying to flog their particular product, but the ones I spoke to, and that was a good dozen or so, were only too happy to sit you down in a quiet corner and go through any of the hardware/software products you were interested in, without even a hint of hard-sell and in some cases, not even a soft-sell. The big surprise was the comments from the show-goers themselves who visited our stand on the upper level, (we and most other user groups were on level 3 of the 3 level show area). They were especially kind and appreciative of the help they got from KAOS on a variety of subjects from micro's to programming tips. We were probably the most unbiased group there. They couldn't believe their eyes when, (on the Saturday), they saw the 'monster' and a few of it's I-nearly-look-as-bad partners being shown in all their glory, with boards, wires and chips hanging out like a hardware hacker's nightmare. I think I got more enquiries and requests for my psuedo German notice than Ray Gardiner got for his brilliant Rabble.

Over the years, I have heard many raves about a particular implementation of high-res graphics on various micro's, but there were two displays at the show that were real eye bogglers. One that most people would expect to be good was the HP. This rather expensive-for-a-micro computer, had a continous demo running that was trific. One part of the demo started with the drawing of a house and tree on a dark moonless night. Dark storm clouds rolled in, thunder clapped and lightning flashed. Each flash of lightning highlighted the side of the house and tree, throwing shadows across the dark green landscape, with incredible speed for hi-res graphics. A window would be lit from within as if an inside light had been turned on.

But the best display I saw was from a system I and and few others had heard of. A 512K, 10Meg hard-disk, 80186 system, punching into a 20" Hi-res RGB monitor. It was creating pictures of photographic quality at an unbelievable rate. The exhibitors were Microprocessor Applications Pty Ltd, and this particular system was just $10,000 complete. When I think of the 15,000 tax-free dollars my boss spent on a Dec Rainbow - with 10M hard-disk and limited software, no graphics mode, where is the video ram - I don't know, how do you peek the keyboard - I don't know, system, I feel like crying. Bye for now.

AUSTRALIAN OTHELLO OPEN

We are searching for the Australian Othello representative who will compete with international Othello champions for the 1984 title due to be held in November 1984.

The first round of the competition has been designed to be very simple, however the complexity of the competition comes in round 2 when the computer issues more difficult Othello puzzles for the competitors. It is round 2 and 3 that the computer enthusiasts should enjoy.

We are offering over 400 prizes so there is a real possibility of winning something. Prizes include a trip for 2 to Honolulu and 6 TAA flights.

Othello's greatest strength seems to be the fact that it can be played at any level, from the youngest starter to the oldest player; it is fun for the whole family to play together. Computer buffs would find the challenge of the Othello competition stimulating and lots of fun. We are expecting to find our best challenges for the Othello title to come from computer people. The rules are straight forward to learn though the game itself might take a lifetime to master. There are Othello Clubs and Associations springing up all over the world and the Australian Association is about to jump!

Please find entry forms included in this issue of KAOS magazine, if further entry forms are required, contact KAOS.

Kerry Whelen
for The Australian Othello Association Inc.
Incorporated under the Associations Act 1981
as a non profit Association.

NB Please forward completed entry forms to:

NOTE As the competition closes on September 6th, interstate members requiring extra entry forms should photocopy the one enclosed.

Superboard

© August 1984
Newsletter of the Ohio Superboard User Group, 146 York Street, Nundah, 4012.

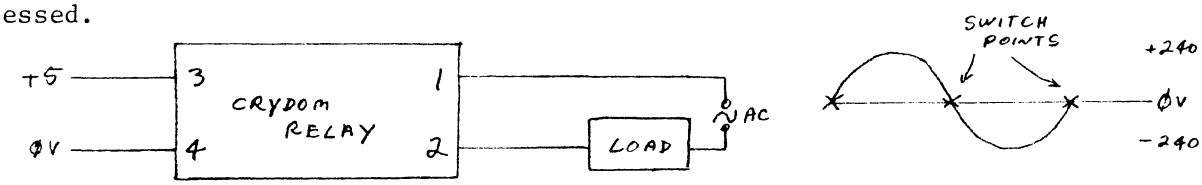
CONTROLLING A.C. DEVICES

One of the possible applications for a computer is the control of AC devices, motors, heaters, lights and etc. To interface these mains powered items safely, good isolation between mains and computer is a must. Another desirable requirement is that the switching device should not cause noise, audible or electrical. Electrical noise interferes with the trouble free operation of computers, and also can annoy radio and television audiences.

An ideal device is a Crydom Solid State Relay, made by International Rectifier in a big range of voltage and current ratings. A D2402 model can handle 240 volts at 2 amps. Any DC voltage between 3 and 32 can be used to turn it on. Connection is very simple as the diagram below shows, and the isolation is 2kV.

Inside the relay is a triac and zero crossing detector. This latter circuit is used to switch the triac on and off a hundred times per second, at the point where the voltage and current flowing is at a minimum - meaning no electrical interference, and also useful for heater element control by duty cycle.

The most obvious use for such a circuit is the control of disk motors on OSI 8" drives which run continuously at 110 volts. Control could be as simple as a toggle switch or a microswitch fitted to the drive door. Alternately, a small circuit could be devised to select motor-on whenever the drive was accessed.



- -

SOFTWARE REVIEW - X-Wing Fighter

X-Wing Fighter is a shoot-em-down arcade game, written in Basic, and requiring 8k. The player is a rear gunner in a spaceship, and the mission is to destroy the enemy X-Wing fighters. What you see on the screen is some stars and your gunsight. On the bottom of the screen are the number of fighters hit, the number that have escaped, and your remaining ammunition. You also get messages like "Here comes another one!".

To steer, you use keys 1 to 8. To make the game extra difficult, the enemy dodge around a lot. Being the rear gunner, the enemy moves in the opposite direction to your ship. (A little renumbering of the steering map will get over the latter annoyance) You can only destroy an X-Wing with a direct hit.

The game is fairly slow and frustrating to play. With a cycle refresh time of about half a second, it's also jerky, and the delay between hitting the spacebar and the phaser firing is devastating. All you can do is keep the enemy near the centre of the sights and blast away, hoping that he'll jitter in the right direction and get blasted. The enemy fighters don't even fire back at you. If you take too long to pin one down, he escapes, and you return in disgrace to the Federation.

Not a lot of fun. The OSUG library has one for members to try out at the usual postage rates.

- -

Next Month:- Some useful information for disk users. Another review.
More changes to CEGMON in the disk bootstrap area.

— SUPERBOARD —

CEGMON MONITOR

CEGMON is a replacement EPROM for the old SYNMON in the C1P/UK101 computers. Developed in the UK in 1980, and sold in the US as the C1E by Aardvark and others, it offers many extra facilities which make the C1P a joy to use. A brief summary appears below, together with information on useful mods to enable it's use for almost any screen/keyboard format.

Features:-

(1) A screen editor very similar in operation to the one in Dabug 3, which makes reworking of Basic lines very easy. Insert, delete, copy, and concatenate with auto-repeat on all functions.
(2) A powerful screen management system. Easily controlled windows provide mixed text and graphics, protected non-scrolling areas, mixed scrolling and non-scrolling areas, full cursor controls, and selective screen and window clears.
(3) A machine code monitor which greatly simplifies programming. Offers a tabular display of hex, facility to input ascii text directly, memory block move, breakpoint facility, and M/C SAVE routine.
(4) A keyboard with proper operation when shift lock is up, just like a standard typewriter. Both shift keys decoded the same, RETURN is always return and etc.
(5) Highly compatible with the standard monitor. A single POKE will enable use of the old $BF2D routine. All major subroutine entry points are identical. Inputs and Outputs are all vectored through RAM.

- -

CHANGES TO CEGMON FOR OTHER FORMATS

The table below gives changes to adapt CEGMON for most other formats in use.

ADDRESS	24x24	32x48	32x64	C4Keybd	Reasons
F819	20	40	40		
F919	F5	F5	E6		T Command 16 bytes instead of 8
F91C	08	10	10		
FB81	20	40	40		
FB8B	D4	D8	D8		
FBBC	17	2F	3F		
FBBD	85	CC	80		
FBBF	85	CC	80		
FBC0	D3	D7	D7		Window set up table
FBC2	85	CC	80		
FBC5	85	CC	80		
FBCB	85	CC	80		
FDF0	20	40	40		
FE2D	20	40	40		
FE3B	D3	D7	D7		
FE62	04	08	08		Pages for screen clear
FFE0	85	4B	40		
FFE1	17	2F	3F		For old $BF2D screen handler
FFE2	00	01	01		
FB4C	FD			02	
FB55	F0			D0	Spacebar detect
FB9A	FE			01	
FBA1	70			50	
FBA4	FB			04	CTRL C detect
FBAB	70			50	
FCBF	FF			00	
FCC4	FF			00	Keyboard non-inverted
FCD3	FF			00	

CEGMON could very easily be adapted for any screen format including 25x80.

CEGMON ENHANCEMENTS by John Froggatt

After I adapted the standard CEGMON to suit my C1 with TASAN board, I set out to make some improvements. The use of a C4 keyboard frees up a small area which can be used to fix that OM ERROR after a warmstart - permanently. Changes:-

```
FD06  20 BE FC    change to  8D 00 DF  )
FD78  20 BE FC      "    "   8D 00 DF  ) In keyboard routine
FF32  4C 00 00      "    "   4C BE FC    In reset routine
```

```
FCBE  A2 FF      LDX #$FF
FCC0  86 88      STX  $88
FCC2  9A         TXS
FCC3  6C 01 00   JMP ($0001)
```

Reset the stackpointer and jump to warmstart. Location $0000 is free.

The next change was to the editor. After nearly a year I found I was still pressing the wrong keys to steer the cursor around. I decided on a more logical arrangement:-

```
FAF0  04     change to   13     Right now CTRL S
FAF4  13       "    "    17     Up now CTRL W
FAF8  06       "    "    1A     Down now CTRL Z
```

Left remains CTRL A and the above arrangement seems easier to remember. I was never satisfied with the interrupt vectors being in the middle of the stack, and decided to move them to page 2:-

```
FFFA  30 01  change to  35 02     NMI now at $0235
FFFE  C0 01    "    "   38 02     IRQ  "  "  $0238
F954  BF 01    "    "   37 02     CEGMON breakpoint routine
```

EXMON needs to be changed to suit the above :-

```
080E  C0 01  change to  38 02
0814  C1 01    "    "   39 02     EXMON breakpoint routine
081A  C2 01    "    "   3A 02
```

At this point, we leave the disk users. Since I bought a BBC micro, I am unlikely to upgrade the C4 to disk. The CEGMON disk bootstrap is from FC00 to FCA5, and I wanted a listing halt routine:-

```
FC95  48         PHA              Save character in accum
FC96  A9 02      LDA #$02
FC98  8D 00 DF   STA  $DF00       Keyboard address
FC9B  AD 00 DF   LDA  $DF00       Is key a spacebar?
FC9E  29 10      AND #$10
FCA0  D0 F4      BNE  $FC96       Yes!
FCA2  68         PLA
FCA3  4C 9B FF   JMP  $FF9B       CEGMON output handler
```

To switch this into the output routine, you change the table copied by CEGMON into page 2 on a reset(BREAK):-

```
FFB4  9B FF    change to  95 FC
```

Now, whenever the spacebar is pressed during a listing it will halt while the spacebar is held down. This has a disadvantage that the auto-repeat on spacebar will be lost. It can be regained with POKE 538,155:POKE 539,255. I have EXMON in EPROM at E800, so the next change gives an E/C/W/M? menu. Selecting E will jump to EXMON instead of the disk bootstrap.

```
FF21  and  FCF8   44  change to  45     E in place of D
FF26              FC    "    "  E8     Jump to EXMON
```

The only bug I have found with the above occurs if you try to Warmstart after using EXMON. The machine used to find it's way back into EXMON but now it hangs. Even this problem disappears with the use of the OSUG BASIC 4 chip.

Some months ago there was a request for help from a member who was using a Tandy printer and having problems with double line feed. Two members sent fixes, the first was in last months newsletter, the second is printed below.

CARRIAGE RETURN TRAP ROUTINE

by Warren Lane

My system is a converted C1, now a C4 (almost!), so I cannot guarantee that the C1 Mini-Floppy version of this routine will run, as I cannot test it with my C4 DOS.

TERMOT or terminal output routine resides at $24CD for both C1 and C4. This is the serial (ACIA) or BASIC Device 1 driver. Unfortunately, there is not room to accomodate any additional code within this routine, so a JSR at $24D6 allows us to fit our trap at any location we desire. On examining a part of the C1 DOS routine, it appears there is a string of zeros prior to the serial driver, so this would appear to be a possible location for the code. If you have a complete disassembly, you may prefer to place the code elsewhere. (Don't forget to change the JSR address at $24D6.)

In operation the routine checks for the presence of a CR rather than a LF, otherwise WP 6502's page feed does not work, since this uses the LF character to achieve this. It appears that if the Tandy VII receives a LF character, it adds a CR, and conversely adds a LF if it receives a CR.

I have included source code for C4 and proposed C1 traps, the C4 routine written over BASIC Device 8 routine, on the premise that it is not used. The most convenient method of entering this code is via a few line of BASIC included within BEXEC*, so these have been included.

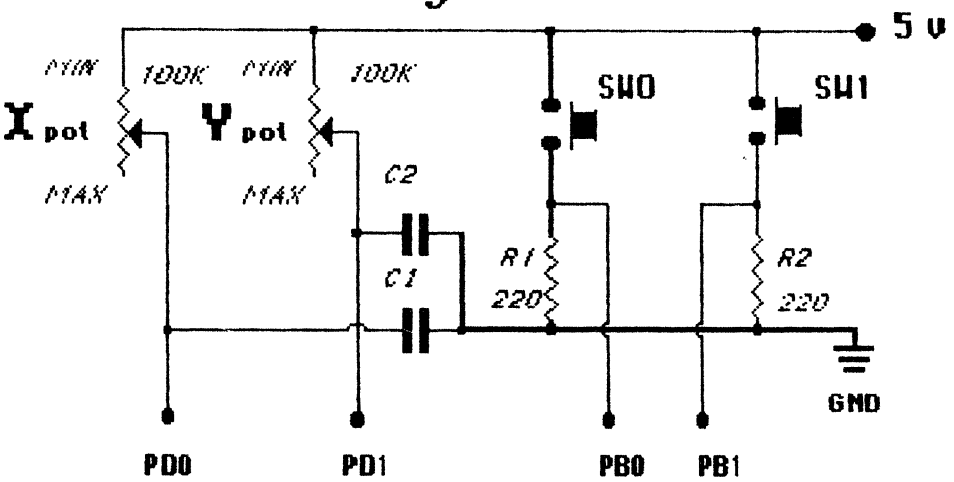
```
10                ;       C1 VERSION
20  24C5=                 TRAP = $24C5
30  24C5                  *= TRAP
40                ;
50  24C5 C90D             CMP #$0D
60  24C7 F003             BEQ RETURN
70  24C9 8D01F0           STA $F001
80  24CC 60       RETURN  RTS
90                ;
100 24D6                  *= $24D6
110               ;
120 24D6 20C524           JSR TRAP
130                       .END
```

```
10                ;       C4 VERSION
20  24B2=                 TRAP = $24B2
30  24B2                  *= TRAP
40                ;
50  24B2 C90D             CMP #$0D
60  24B4 F003             BEQ RETURN
70  24B6 8D01F0           STA $F001 **
80  24B9 60       RETURN  RTS
90                ;
100 24D6                  *= $24D6
110               ;
120 24D6 20B224           JSR TRAP
130                       .END
   ** NOTE: $FC01 for true C4.
```

```
1 REM LINE NUMBERS MAY REQUIRE CHANGING TO FIT BEXEC*
2 REM C1 DOS VERSION BUT NOT TESTED
25 POKE943Ø,32:POKE9431,197:POKE9432,36
3Ø FORX=9413TO942Ø:READA:POKEX,A:NEXT
35 DATA 2Ø1,13,24Ø,3,141,1,24Ø,96
```

```
1 REM LINE NUMBERS MAY REQUIRE CHANGING TO FIT BEXEC*
2 REM C4 DOS VERSION
25 POKE943Ø,32:POKE9431,178:POKE9432,36
3Ø FORX=9394TO94Ø1:READA:POKEX,A:NEXT
35 DATA 2Ø1,13,24Ø,3,141,1,24Ø,96
NOTE: For a true C4 - DATA 2Ø1,13,24Ø,3,141,1,252,96
```

APPLE JOYSTICK



Constructing an Apple joystick.

Apple joysticks are expensive to buy, but cheap to make yourself. Especially if you are only after a joystick to play games with, as accuracy is not as important as it being rugged and cheap.

In keeping with these requirements I chose to use the joystick mechanism which is readily obtainable from a Tandy or Dick Smith store, and costs about $5.00. The mechanism comes with built in 100K pots and is equipped with a small lever which although not self-centering; is light and easy to operate. While you are in the store getting the joystick, buy a small plastic jiffy box to hold it. You will need one 10x5.5x4 cm in size to hold the joystick mechanism and the two Fire buttons.

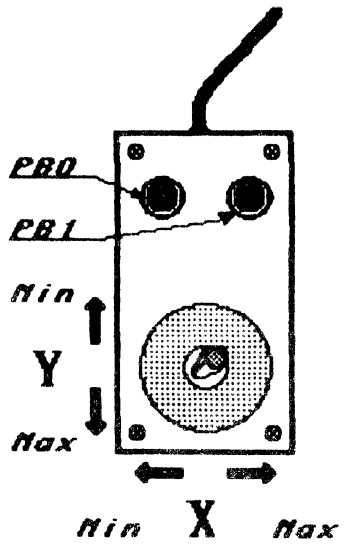
The Fire buttons, are in my opinion, the most important; they must have a light action and short travel as there is nothing worse than being half way through an Alien attack when your thumb falls off due to the 10 ton pressure required to fire your cannon.

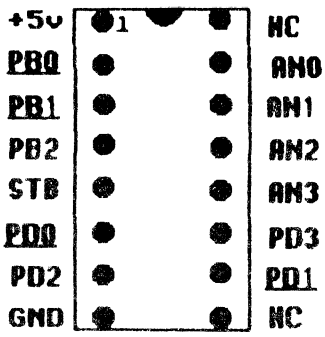
The best firing buttons are either those made using miniture micro swiches or the C&K 8121 push button switch. Unfortunatly neither Tandy or Dick Smith sell the C&K switches, but Magraths, Radio Parts, Ellistronics, and Rod Irving do. You will also need the Plunger buttons to go with the 8121 switches. The total cost of each switch is about $3.50, not cheap, but well worth it.

You will also need 1.5 metres of 6 core cable or ribbon cable and a 16 pin dil header to plug into the Apple Paddle connector. The best ones are those sold by Magraths, they're made by Camdon and have a fiber glass body and therefore don't melt when you solder them. They cost about $1.75.

The circuit itself is straight forward but the values of C1 and C2 are not. The problem is that the Apple joystick feeds a 558 timer which expects the X & Y pots to cover a range of 0 to 150k ohms. As our pots only cover 0 to 100k we have to increase the time constant the timer sees. As the time constant varies slightly from machine to machine and the tolarance on the joystick pots is about 20%, you are going to have to select the values for C1 & C2. With the joystick assembled but no C1 or C2 fitted, run the test program; you should get a range of 0 to 255 for the full travel of the stick in either X or Y. Without the capacitors you will probably only get to 200 so, with a pair of clip leads, clip a 0.01 ufd capacitor between the wiper of the pot you are testing and GRD. If it still does not get there, then increase the value till it does. If on the other hand it reaches 255 with the stick still only part way through its travel, then reduce the value.

When you have finished you should have a stick that goes from 0 to 255 in both directions, with the stick returning a value of approx 127 when the stick is vertical. Solder your capacitors into circuit, assemble the rest of the box and go play.

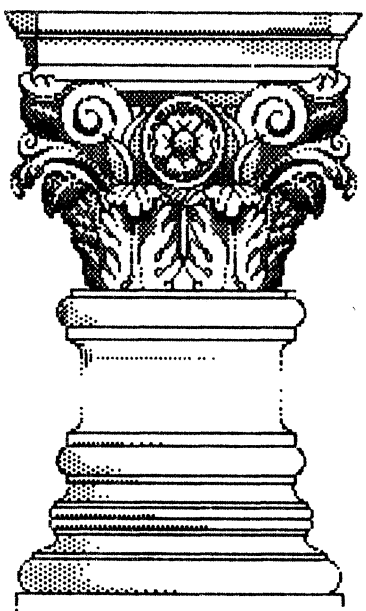




GAMES I/O CONNECTOR

```
10 REM JOYSTICK TEST
20 HOME
30PRINT"X VALUE = ";
 PDL(0):PRINT
40FORX=1 TO 10: NEXT
50PRINT"Y VALUE = ";
 PDL(1):PRINT
60B1=PEEK(-16287):
 B2=PEEK(-16286)
70 IF B1 >127 THEN A$
 ="CLOSED"GOTO 90
80A$="OPEN"
90 IF B2 >127 THEN B$
 ="CLOSED":GOTO110
100B$="OPEN"
110PRINT"FIRE1=";A$;
" FIRE2=";B$:GOTO20
```

By the way, this article was written on the Appie Macintosh using MACPAINT for the diagrams, and MACWRITE for the text..... D. J. Anear.
Here is a sample of some of the Graphics that can be done on the MAC.



GET SMART JOIN KAOS

MEETING KAOS SYDNEY

By Norman Bate

Since my last report of our April meeting we have had only one other in June. Attendance is still low and is disappointing. Our April meeting attracted 8 members and 4 computers while the June meeting saw 7 members and 3 computers.

Peter Jensen used his amateur radio gear to try and arrange a link to the Melbourne meeting. Contact was established but conditions were not good and the signal only poor to fair, barely reasonable for voice but out of the question for data exchange. Peter is going to try again during the August meeting.

Several of our members are experimenting with 'VEKTRIX' games units to see if they can be used as graphics peripherals. More on this later. CP/M machines are beginning to surface.

The next bi-monthly meeting will be at the Lugarno Girl Guides Hall on Sunday August 26th any time after 10 am. For information contact:
N. Bate or N. Bissett

UNDERSTANDING FORTH Part 2

by Ray Gardiner

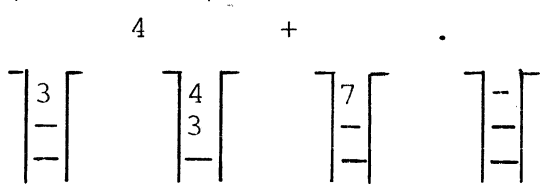
This month we will examine the two stacks in the Forth system and explain the details of stack manipulations.

: PARAMETER STACK

All parameter passing and arithmetic calculation takes place on the parameter stack, usually just called the STACK.
Before we begin to explore stack operators in detail a description of the stack is required so that we may visualise what is taking place.

The Forth stacks are both LIFO (last in first out) stacks and only the top item of the stack is visable directly at any given time. Data is returned in the reverse order to the order that it was entered (last in first out).

A LIFO stack may be pictured as a spring loaded device (Leo Brodie has some nice cartoons for this).
If we type 3 4 + . <cr> what happens? Examining the activity pictorially as the input is interpreted.



At this point the stack is returned to the same state as it was initially.
You should make sure that your definitions do not leave rubbish behind.

: DESCRIPTION

When 3 is encountered the interpreter pushes 3 onto the stack after first searching the dictionary for a word called "3".
Assuming that no definition is found, interpret then calls a word called NUMBER which attempts to convert the ASCII character "3" into a binary number. If this process succeedes the binary number is left on the stack. The same thing happens with the "4". The next delimited character found from the input stream by the interpreter is "+" which is a standard Forth word defining the 16 bit signed fixed point addition process. (Adds 'em together dummy !). Remember that the number on the stack is in binary after NUMBER. Finally the Forth word DOT "." takes the top of stack and converts it to an ASCII string and outputs the string to the terminal.

Definitions of INTERPRET NUMBER and . will be given in the section dealing with the interpreter. It is important to note that the string conversions for NUMBER and . both utilize the system variable BASE - more on this shortly.

: STACK MANIPULATIONS

In order to describe stack effects of various operators we need to introduce the standard stack notation system used in documenting Forth words.

Word	Pronounced	Stack Before	Stack After	Comments
DUP	"dupe"	w1	w1 w1	copy top of stack
SWAP	"swap"	w1 w2	w2 w1	reverse order
OVER	"over"	w1 w2	w1 w2 w1	copy second item
ROT	"rote"	w1 w2 w3	w2 w3 w1	rotate top three
DROP	"drop"	w1	...	destroy top

: STACK NOTATION

Stack Abbrev.	Number Type	Range in Decimal
flag	boolean	Ø=false 1=true
b	bit	Ø or 1
char	character type	Ø to 27
8b	byte	
n	number signed	-32768 to +32767
+n	positive number	Ø to +32767
u	unsigned number	Ø to +65535

: STACK NOTATION

Stack Abbrev.	Number Type	Range in Decimal
w	signed or unsigned (n or u)	
addr	same as u, usually addresses	
d	double number	-2,147,483,648 to +2,147,483,647
d+	signed double	Ø to +4,294,967,295

:STACK ARITHMETIC

Word	Pronounced	Stack Before	Stack After	Comments
+	"plus"	n1 n2	n1+n2	Add top two items
-	"minus"	n1 n2	n1-n2	subtract top two
*	"star"	n1 n2	n1*n2	multiply top two
/	"slash"	n1 n2	n1/n2	divide top two
MOD	"mod"	n1 n2	n3	leave remainder
/MOD	"slash mod"	n1 n2	n3 n4	leave remainder & quotient
MIN	"min"	n1 n2	n3	leave smallest
MAX	"max"	n1 n2	n3	leave largest
ABS	"abs"	n1	u1	absolute value

: EXAMPLES

1. Calculate the temperature in degrees F given degrees C on the stack.

```
: C->F  9 * 5 / 32 + ;          (degrees C --- degrees F)
```

Testing this definition is simply a case of entering a range of numbers to verify that the correct calculation is being carried out.

```
15  C->F  . 59   OK
1ØØ C->F  . 212  OK
```

2. The height above ground of a falling object can be determined at any time (t) if it's initial height (ho) and initial downward velocity (vo) are known. The height of the object is given by the formula

height = ho - (vo*t) - 32 t*t

where height is in feet and velocity is in feet per second. We can write a Forth definition which will take an initial height, initial velocity, and time t, from the stack and leave the final height of the object after t seconds.

```
: HEIGHT  SWAP OVER * SWAP DUP * 32 * + - ;
```

initial height 1ØØØ ft, initial velocity 1ØØ ft/sec, how high after 3 seconds?

```
1ØØØ 1ØØ 3 HEIGHT . 412  OK
```

The detailed stack manipulation can now be examined step by step.

Word	Stack	Comment
	1ØØØ 1ØØ 3	ho vo t
SWAP	1ØØØ 3 1ØØ	swaps time and velocity
OVER	1ØØØ 3 1ØØ 3	copy time to top of stack
*	1ØØØ 3 3ØØ	calculates vo*t
SWAP	1ØØØ 3ØØ 3	get time back to top
DUP	1ØØØ 3ØØ 3 3	prepare to square t
*	1ØØØ 3ØØ 9	calculate t*t
32	1ØØØ 3ØØ 9 32	gravity enters the scene
*	1ØØØ 3ØØ 288	calculate 32t*t
+	1ØØØ 588	ho (vo*t+32t*t)
-	412	ho - (vo*t+32t*t)

The result is left on the stack for the next word.

A cleaned up and polished version of HEIGHT is now presented, although the basic word is the same the presentation is perhaps a little clearer.

```
: HEIGHT?  CR ." Enter initial height ?  " QUERY INTERPRET
           CR ." Enter initial velocity? " QUERY INTERPRET
           CR ." Enter time delay        " QUERY INTERPRET
           CR ." After " DUP . ." seconds "
           CR ." the object will be " HEIGHT  Ø MAX
              ." feet above the ground " CR CR ;
```

Note the Ø MAX after HEIGHT so that when an object hits the ground it stays there.

: RETURN STACK

The return stack is used for holding the return address of the calling Forth word, as such it is important that it is not unwittingly corrupted, or you will find yourself wondering what happened to the system.
The return stack is also used to hold loop indices for some of the Forth control structures, more on this when we look at control structures next month. For now we will consider the word which allow the programmer to make temporary ue of the return stack. However be careful.

```
>R  " to R  "  moves top of parameter stack to return stack
R>  " R from"  moves top of return stack to parameter stack
R  " R "    copies non-destructively top of return stack to parameter stack
```

: USING THE RETURN STACK FOR TEMPORARY STORAGE

The use of R> must be balanced by >R within the definition, failure to do this will result in a system crash (usually). Some Forth words actually manipulate the return address to achieve certain results, like (.") which manipulates the return address to step over an embedded string. Also words like EXIT which discard the return address can be used to control the system.

```
: EXIT R> DROP ;  (what could be simpler)
```

REVERSE POLISH NOTATION VERSUS ALGEBRAIC

One of the hardest things for a beginner in Forth to grasp is the post-fix notation which seems to pervade the language. From our earliest days in the classroom we are taught that 1+1=2 is the natural order of the world and now Forth is telling us that inserting the operator "+" cannot be logical until we have something to operate on. All, repeat ALL languages must translate from algebraic notation to reverse polish before any code can be executed. A direct result of adopting INFIX is the need to establish the operator precedences, remember BOMDAS? Consider the following example.

```
6 X 5 + 11 - 3
```

There are a multitude of answers to the value of the above expression depending on which operator has precedence as decided by the positioning of some brackets which will settle the issue of what to do first. Before someone takes me to task and points out that "X" has higher precedence than "+" and the answer is 38, the POSTFIX requires no confusing ideas of what to do first, the order is specified by the order on the stack. ie.

```
6 5 * 11 + 3 - . 38      6 X 5 + 11 - 3
or
6 5 11 + * 3 - . 93      6 (5 + 11) - 3
```

At the very heart of this issue is the arbitary decision to teach infix notation until we all succumb to the false belief that infix is the natural order of the world.

During this series of articles I am going to refrain from any attempt to categorise Forth or compare it to other languages, there is far too much involved to even try.

: EXERCISES

1. Convert the following into postfix notation
   (i) k1*a*a*a + k2*a*a + k3*a + k4
   (ii) (a+b) / (a-b)
2. Write a Forth word to calculate the points difference for a football match where the scores are in goals and behinds ie. 11 13 9 25 ?WIN returns the winning/losing margin.

: ANSWERS TO PART 1 EXERCISES

1. `: Lance ." Leventhal " ;`
2. `: Compsoft ." 428-5269 BH " ;`
3. `: merci ." thank you " ;  : pour ." for " ;  : fleurs ." flowers " ;`
   `: le ." the " ;` merci pour le fleurs ---> thank you for the flowers

Firstly, I must apologise to those people who turned up to the meeting according to the date I mentioned in my previous report (a Saturday) or on the following Sunday. I must have been looking at the wrong month when I worked out the date.

The last meeting (which never the less was well attended) did take place on Sunday 15/7/84. Apart from the usual sharing and discussion which took place, a decision was made to hold a more informal meeting (if that is possible) on the third Sunday of every month at 2.ØØ pm. This came about to help our newer members and for those people who developed problems and would like to talk to others, other than at our usual bi-monthly meetings. These new meetings are to take place at the home of Gary Giles,

The first meeting will take place on Sunday 19/Ø8/84. It will possibly already have occured when you receive this newsletter.

Our mext meetings are:

1. Sunday 19 aug. 2.ØØpm at Lot 74 Murchison Way, Gosnells (cnr DeGrey Cl)
2. Sunday 16 Sept 2.ØØpm at Guild House, 56 Kishorne Rd, Mt Pleasant
3. Sunday 21 Oct. 2.ØØpm at Lot 74 Murchison Way, Gosnells (cnr DeGrey Cl)

Gerry Ligtermoet

ADAPTING AN APPLE 8Ø COLUMN CARD TO OSI

by Darryl Lock

Apple 8Ø column card copies can now be purchased for less than $1ØØ.ØØ, so I decided to adapt one of them to suit OSI (or any memory mapped computer) rather than build one from scratch myself. Most 8Ø column cards use the 6845 CRTC chip, which means many screen formats are available. I have used 8Øx24, 64x32, 48x12 and 24x24 successfully so far and a 132 column screen should be possible.

Adapting this card to OSI is easy enough but it should only be tackled if you have a reasonable knowledge of hardware and can find your way around a Rabble expansion board well enough to locate the necessary signals for the Apple edge connector.

The card that I purchased is a copy of the Videx Videoterm card, it is available from Australian Video Presentations in Melbourne. The Videoterm card features its own clock (17.43 MHz), 2K of 2114L RAM, room for expanded character sets, reverse video capability, light pen (you have to build your own pen), a character set which includes a comprehensive line drawing set and special characters to allow 'chunky' graphics of 8Øx72 (in the 8Ø column mode) and characters for 'display function' as is available on some terminals (ie. characters for <cr>, <lf>, <bs> etc.). The display is clear, steady and uses well formed characters.

In order to use the card with my OSI C1P with Rabble expansion board, I installed an edge connector socket in the Rabble prototype area. The socket is a 5Ø way, Ø.1" spacing type as used for Apple I/O slots. It is not the same width as a chip, but the pins can be bent far enough to go through the holes in the prototype area. The card mounts at right angles to the Rabble exp. board, so if you have your board in the same case as your Superboard or whatever, it probably won't fit and you will have to devise another method of mounting.

As my main reason for installing this card is to convert to a C4 lookalike and I want to use software available for the Rabble machine, I also installed 2K scratch RAM at $C8ØØ. The decoding for this and the 2K video RAM on the Videoterm card can be achieved with one 74LS139 chip in the prototype area (see circuit). The necessary signals to drive the video card are: GND, +5V, RESET, DEVICE SELECT, Q3, DØ-D7, AØ-A1Ø, I/O STROBE and R/W. All will be

familiar except Q3 which is an Apple signal used to enable writing only during phase two clock when addresses have stabilised. The circuit used for the decoder uses phase two so this one (Q3) can simply be grounded. Be careful with the other enables which uses negative logic, DEVICE SELECT, I/O STROBE and RESET are all active low. R/W is low for the write cycle. Use buffered signals if at all possible.

Now a word about Apple I/O slot conventions. Apple use three select signals to operate the slots. The first is called Device Select, and it allows access to hardware and the registers of the CRTC chip. The output of decoder chip IC8 pin 9 on the Rabble expansion board is suitable. It will give the CRTC addresses $CØ1C-$CØ1D (also $CØ1E-$CØ1F but that's not much waste). The second signal is I/O Strobe which, on the Apple, selects addresses $C8ØØ-$CFFF for whichever I/O slot is being used (there are hardware latches to turn each card's $C8ØØ-$CFFF area on and off). With some small modifications to the card I used this as my video RAM select $DØØØ-$D7FF, which comes from the 139 decoder in the proto area. Lastly, Apple use a signal called I/O SELECT to select an on-board ROM, which, as it is filled with Apple screen driver routines, is useless to us.

The 2K RAM on the video card is, unfortunately, paged as four pages controlled by two latches, so I pulled the chip containing the latches (it is labelled U8). Luckily the card is completely socketed, and this is easy to do. With the chip out the two upper address lines of the RAM are free and can be connected direct to the processor address lines which are available on the Apple edge connector. While I was doing this I also removed U3 (the EPROM) to reduce current and possibly to reuse later. A9 and A1Ø can be found on U9, pins 3 and 2 respectively. Connect these pins to U16, pins 5 and 11 respectively. Next remove U24, bend pin 1Ø out so that it won't go back in the socket when you replace the chip, then replace the chip. This allows us to connect to pin 1Ø without cutting tracks. Do the same trick to U2 pin 5. Now unsolder the diode (there is only one) at the positive end and leave it clear of the board. Connect the positive end to U2 pin 5 (sticking out) and U2 pin 6 (still in the socket) and also to U24 pin 1Ø. You have just changed the I/O STROBE into a video RAM select, which will come from the 139 decoder in the Rabble prototype area. (To appreciate the above mess it is necessary to study the circuit diagram.) The circuit diagram in the Videoterm manual is very small, so I don't know how well it will reproduce in the newsletter. If it is unclear I will send multiple copies of my original to KAOS for anyone who needs one. I recommend that you buy the manual (mine was about $9.ØØ) as it contains much information, such as character sets, reverse video options etc., that will be very useful.

If you use C4 software and 64x32 mode, the display works very well and all that is necessary for you to add is a setup routine for the CTRC registers. This can be done as an addition to the startup sequence in EPROM so that you have video when you turn the machine on, or as part of the DOS bootup (from BEXEC* if you like). RESET does not reset the CRTC registers, so that once started, you have video until you remove power from the board. Software to drive 8Ø columns is apparently available for Rabble 65 type computers and for the CP/M system from Compsoft. At present I am awaiting the arrival of my CP/M system which at first I will use with 64x32 format, but as soon as I get it going I will be moving over to 8Øx24. The only item I am not sure about, with regard to complete compatability, is reverse video. The video card uses the most significant bit of video memory as the reverse video bit, if that option is selected. As supplied, the full 256 characters are available with no reverse video. Because 16 locations are used for each character in the character generator ROM, a 2732 is used. The first 128 characters are fairly standard ASCII with some of the non-printables replaced with graphics. The second 128 are only half programmed leaving 64 that you could program yourself. The programmed 64 are line drawing graphics. So, if you choose reverse video you have the first 128 (the ASCII set) available, but you lose the line drawing. There is space on the card to add a socket and another set of 128 characters using a 2716, but this would have to be switched

manually if you use reverse video. You can also select all reverse video, but personally, I don't like it. The blanking when the CPU accesses the video RAM is evident.

Finally, a word on the decoding circuit. The only part that may need comment is the signal to the data direction gate on the Rabble board. The $DXXX switch must be open as screen RAM is now on the Rabble expansion. However the keyboard is still on the main board, so therefore the addresses $D8ØØ-$DFFF on the Rabble must be disabled so no conflict will occur. Those of you who don't use buffers to the Rabble or who have modified the data direction circuit may have to think about what is appropriate for your setup. Also don't forget to pull the video RAM buffers on the main board to avoid conflict there.

Programming the 6845 chip has been very well described in previous issues of KAOS and is also described in the manual for the card, so I won't say much about it here. Follow the examples in KAOS but remember the clock frequency is 17.43 MHz. For 8Øx24 the setup is given in the manual, just turn off the cursor. This is because OSI use a character as a cursor, so you won't need it and you'll lose it anyway if you select reverse video. Some setups are for 6Ø Hz frame and some are 5Ø Hz. Most monitors will accept anything in this range, so don't try to calculate for exactly 5Ø Hz. About monitors, I have seen modified TVs working with 8Øx24 but the result is poor. It can be improved by reducing the video output from the card to as low a level as possible without losing too much contrast. This can be done by changing a resistor divider on the video card. This helps because the TV circuits can respond to small video changes faster than large ones, giving effectively a better frequency response at low levels, so, if white on black is blurred, white on grey may not be. I have an Apple IIe monitor which is excellent, and so would most of the cheap monitors available, provided they are not modified TVs.

I will try to answer enquiries from anyone needing more information. As I live in phone calls will be expensive but my address is

---

Ciruits supplied with this article were not suitable for reproduction in the newsletter. If you would like copies, contact KAOS and we will post them to you

The board mentioned in this article is available from:
Australian Video Presentations, 325 Bank St, Sth Melbourne 32Ø5
Ph Ø3 699 4177 The cost is approx. $85.ØØ plus tax.

HELP WANTED

I have a C4 MF with a 48K Studiotech board and standard disk controller. I also have an 8" disk drive that I want to configure as the B drive and run both 5.25" and 8" drives. Does any member have details of how to do this?
Bob Best,

FOR SALE

C3 OEM OSI triple processor system, 65Ø2, Z8Ø, 68ØØ, 56K static RAM, 8 slot backplane, 2 8" drives, stacks of software on 9Ø 82 disks including CP/M, BASIC, Fortran, Cobol, Forth, Pascal, Macro, plus many hardware extras, OS manuals etc. Worth well over $6ØØØ. Sell for $2ØØØ ONO. Also a GE Terminet hardcopy terminal, 11Ø, 3ØØ, 12ØØ baud, including all operating/servicing manuals. Cost over $2ØØØ, sell $5ØØ. See Ron Cork or call

Registered by Australia Post
Publication No. VBG4212

If undeliverable return to
KAOS, 10 Forbes St
Essendon, Victoria 3040

KAOSKAOS
K Postage K
A Paid A
O Essendon O
S 3040 S
KAOSKAOS